
The New Way of Doing Migration

Maynor Alvarado A

May 2002

GW10502010

Contents

1	Introduction	3
1.1	Background	3
1.2	Current Limitations	3
1.3	Fear of Change	4
1.4	Ideal Panorama	4
1.5	Solution	4
2	Technological Mechanism based on Artificial Intelligence	4
2.1	Informix 4GL-to-Java Migration Tool	5
2.2	Concept Mapping	5
2.3	Compatibility Classes	6
3	Impact of automatic migration on a Real-Life Project	7
3.1	A Real Life Project's Comparative Data	7
3.2	Automatic Migration: Real Implications	8
4	Conclusion.....	9

The New Way of Doing Migration

1 Introduction

Well designed, legacy applications pose a constant technological challenge for IS divisions everywhere. We have invested invaluable human and financial resources in these systems and cannot afford to abandon them due to the potential loss of the proven knowledge capital embedded in them. We wish to take better advantage of the resources we have invested in order to offer our clients, suppliers, employees and stakeholders the highest caliber of technological services. Unfortunately, the technological environment that handles our business processes and know-how is lagging so far behind any business reality that it has begun to limit our applications' potential to support, let alone enhance, our organizational strengths in an ever more changing and competitive world.

1.1 Background

I believe that as Managers or those in charge of Information Systems, the word conversion or migration has struck us at one time or another during our professional careers. This is particularly so when we have to confront certain problems with the technology that our company is using. These problems could be due to technological obsolescence, a sub optimal service we are getting from a supplier, technological constraints in our systems, shortage of expert resources in our environment and so on.

1.2 Current Limitations

It is very simple to think of an example that clearly shows the type of problems we have been mentioning. Suppose we have a mission critical application that has been developed in Informix 4GL, a practically obsolete programming language, i.e. it has not evolved significantly over the past 10 years. Our technical staff in charge of maintenance has been resigning as they have lost interest in the tool because of its obsolescence, and they naturally wish to avoid becoming obsolete along with the tool. As such, they are seeking new horizons in the labor market that would enable them to be in touch with the latest technologies. In addition to this staffing problem, our company managers have asked us to publish part of our application on the extranet through our information systems so that our suppliers can manage the inventory of their goods in our warehouses more effectively. And to add insult to injury, an increasing number of users have been complaining about our unfriendly systems interface, wondering why our company is so technologically underdeveloped!

1.3 Fear of Change

It is precisely at such a time that we start thinking about migrating away from our system. However, those of us who have participated in a conventional migration process know only too well what the implications of such a migration decision are. The first thing that comes to our mind is a series of problems and fears associated with migration. Such fears could be the excessive cost of a software development project; the time overruns with respect to the project deadline; the similarity between the project's final product and my mission critical application; the need to retrain my technical staff, required training for end users; and lots of other similar factors that would make me reluctant about undertaking such a project.

1.4 Ideal Panorama

It is also at this very moment that we start to wonder about an ideal tool capable of carrying out the main part of our migration work toward a modern language in an automatic fashion; where we would only have to provide our source code, and the only work we would have to do would be to test the migrated application we are receiving in order to make sure it behaves just like the original application. Further more, wouldn't we love to receive an application as the end product of this migration that would enable us to break free from the current chains of inflexibility and dependency that limit our decision-making process?

1.5 Solution

Nowadays, ArtinSoft, a pioneer and worldwide leader in migration services, makes this ideal world become a true reality. Its modern techniques based on artificial intelligence empower us to carry out systems migration with an extremely high degree of automation, evolving the market along with a new way of carrying out migrations.

In this document, we shall explain how this process is feasible and highlight the differences between the traditional approach to migration (manual re-writes) and the way ArtinSoft-Style Migration is carried out.

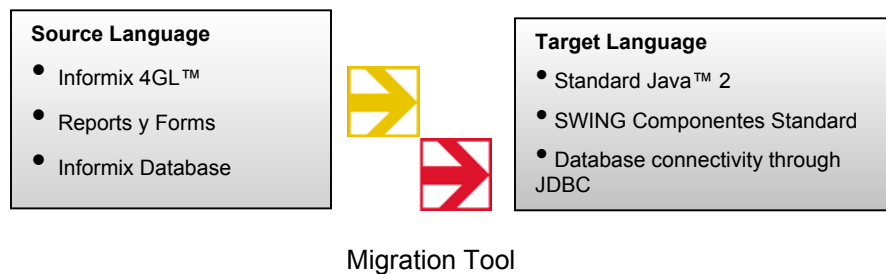
2 Technological Mechanism based on Artificial Intelligence

Through its pioneering migration technology, Freedom, ArtinSoft employs grammar-based reasoning to optimize mission critical applications' migration process. At the core of Freedom there is a proprietary Artificial Intelligence technology that creates a full abstraction of the original program via intermediate representations in which millions of transformations are applied to produce another abstraction in the target language. The new source code is then generated from that abstraction. At each processing stage, the system cleans-up

the logic, derives useful information from it, deduces its new best representational form and passes it on to the next stage.

2.1 Informix 4GL-to-Java Migration Tool

ArtinSoft has created a technology capable of converting an application developed in Informix 4GL to Java. 95% of the source code is migrated automatically - something unheard of in the history of software. The following graph shows what input the migration tool requires and what the outcome of the process will be like.



As you can see, what the migration tool requires in terms of input consists of the source code programmed in Informix 4GL. This must contain the programs, reports, forms and the database structures that are used by the application. Once the input is provided, the migration tool, developed by ArtinSoft, applies a series of conversion rules and generates a source code that is 100% standard Java 2. This source code uses swing components for the graphical presentation of the application and database connectivity through JDBC (Java Data Base Connectivity).

2.2 Concept Mapping

A high percentage of the success of ArtinSoft's migration tool is related to what we have denominated "Concept Mapping"¹. Concept Mapping is about representing Informix 4GL language and structure in Java. This is to say that every component in Informix 4GL is mapped to a Java code equivalent. In order to clarify this point, we shall provide a few examples.

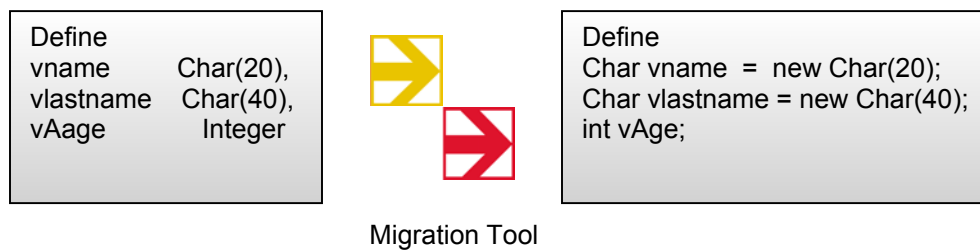
The first and the most basic case can be observed through an example of file names. In Informix 4GL, all the files have the 4GL extension. In Java, on the other hand, all the source code files have the Java extension.

¹ Taken from ArtinSoft Partners Manual



Like file names, the migration tool preserves the structure of directories, names of forms, and names of all objects that together comprise the original application in 4GL.

Now, we shall proceed to analyze the content of the files that are generated in Java. Let's begin with an example of variable declaration:



As you may observe, what changes from one language to another is the representation of declarations, which is straightforward as the way to define variables in Java is different from the way variables are defined in 4GL.

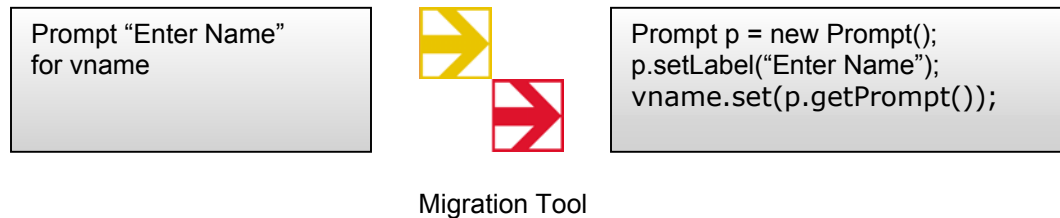
Preserving the name of those objects created by the programmer in the target language is of great help. This is simply because someone who knows the source application programmed in 4GL, though s/he may not be an expert in the target language, i.e. in this case Java, can thoroughly understand the generated program's logic.

2.3 Compatibility Classes

Another interesting point to be highlighted in the example above is a data type called "Char" for which there is no equivalent in Java. However, ArtinSoft has found a solution to this problem through what it has named "Freedom Classes". For this case in particular, an object called Char with the same characteristics and behavior as the char field in Informix 4GL was created in "Freedom Classes". The main difference between the two lies in the fact that the Char object of "Freedom Classes" is 100% programmed in Java.

"Freedom Classes" also contain a representation in Java of those instructions used in Informix 4GL. In order to provide an example, we shall use the Prompt

instruction employed in 4GL to display a message on the screen and request a key input.



This example shows that the representation of Informix 4GL Prompt instruction in Java is very simple and intuitive, reason why an Informix 4GL programmer will easily know how to program a Prompt in Java by using Freedom Classes.

In sum, the main reason behind Concept Mapping and Freedom Classes is to simplify a programmer's job when it comes to giving maintenance to the migrated application. By having the file names, variables, functions, reports, forms and other 4GL objects migrated to Java, and with there being a representation of 4GL instructions in Freedom Classes, we can envisage an ideal situation for the original application's programmer as it would be very simple for him to understand and adapt to the new way of programming, even if a new language, such as Java, is being used.

3 Impact of automatic migration on a Real-Life Project

Taking into account the language table proposed by Capers Jones, one of the scholars in the area of Software Productivity², we now move on to a detailed analysis of a case study related to automatic migration.

According to Jones, a Java programmer's maximum productivity is 1,060 lines a month. This is 260 lines a month more than an Informix 4GL programmer (800 lines a month) can manage. Jones states that the number of lines per month is not directly related to the number of lines that a programmer is capable of codifying. This aspect is more related to several factors such as requirements analysis, the time it takes to understand the problem, the time needed for testing, and the time it takes to make sure that the final product is stable. All these factors mean that only 30% of the productivity is directly related to programming.

3.1 A Real Life Project's Comparative Data

Under the above perspective, we can analyze a case study based on a project that ArtinSoft executed for one of its customers called Mundo de Juguetes³. This

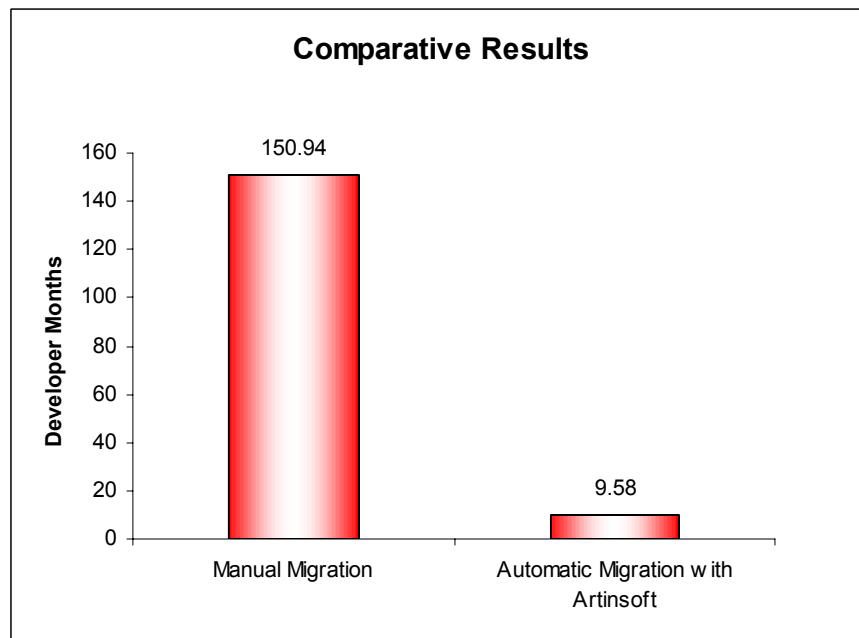
² Programming Languages Table Release 8.2, March 1996. Carper Jones, Chairman, Software Productivity Research, Inc. © Copyright 1997 by Software Productivity Research, Inc. All Rights Reserved.

³ For detailed information, refer to [Study Case: Mundo de Juguetes](#)

basically consisted of migrating one of their mission critical applications, Point of Sale System, from Informix 4GL to Java, as well as changing their database from Informix Online 5.0 to Oracle 9i.

The standard Java 2 code for the automatically migrated Point of Sale application has a total of 160,000 lines. Using the statistics from Jones' study, 150.94 programmer months would have been needed to reprogram the application manually, test it and make sure it is stable before it is ready for production. This means that for it to be considered a viable project in terms of time, 6 programmers working for a period of 2 years would have been needed. This naturally assumes that the project would be handled to perfection, with no major problems of any sort during the development period

Thanks to ArtinSoft's automatic migration technology, only 9.58 programmer months (2300 programmer hours) were considered necessary to complete the project and have it ready for production. In the real project, a team of 3 programmers worked for a period of 3.19 months to have the application running productively in the company's 12 nationwide stores. This is less than 6.35% of the time it would have taken to migrate the whole project manually.



3.2 Automatic Migration: Real Implications

It is at this stage that some of us may wonder about a critical question: Why are we talking about automatic migration if it took 3 developers 3.19 months to fully convert the application and have it ready for production? Actually, the answer is quite simple, though before we continue, it is important to point out that automatic migration refers to a process with minimal human intervention. Do not confuse

automatic with “instant” processes such as making coffee, a simple process as it may be, it does not produce the same taste or flavor as real coffee, anyhow! The first reason is that it is impossible to achieve 100% functional equivalence when migrating an application from Informix 4GL to Java, mainly because of two factors: the paradigm shift between the two languages, and the dependency of the Informix 4GL developed application on the platform where it is executed. The second reason has to do with the knowledge or work that has to be developed or provided by human beings. A clear example of this occurs at the stage where the system is tested. These tests cannot be carried automatically without any user intervention. The users are the only ones who know how the system behaves and what the results that the system produces should be like.

The company’s Point of Sale system was put to work in Java and started running on an Oracle 9i database successfully. An important result of this process was the fact that no end user training was required as the project’s goal of reaching 100% functional equivalence was fully attained

4 Conclusion.

Thanks to breakthroughs in technology, we are now able to enjoy a completely different panorama. A clear panorama that provides us with a proven and viable solution to all our traditional problems of saving and maximizing the full potential of knowledge capital embedded in our mission critical applications. This pioneering solution enables us to free ourselves from the chains of obsolete technologies, and also from hardware and software suppliers who tend to make our panorama uncertain with temporary and/or imperfect solutions.

No only has ArtinSoft created a migration tool that enables us to automatically convert more than 95% of our Informix 4GL developed application to Standard Java, it has also developed a new approach to migration. ArtinSoft has created a completely new methodology that enables us to standardize language conversion process as well as control the various variables that affect our software development projects. We can thus conclude by saying that we definitely stand before “a new way of doing migration”.

ArtinSoft© 2002. All rights reserved.

This document may be reproduced subject to the condition that it is done in a total fashion, without any modification to the form, and by imposing and maintaining this note in all subsequent copies